Online Algorithms with Discrete Visibility

Exploring Unknown Polygonal Environments

©DIGITAL VISION

The context of this work is the exploration of unknown polygonal environments with obstacles. Both the outer boundary and the boundaries of obstacles are piecewise linear. The boundaries can be nonconvex. The exploration problem can be motivated by the following application. Imagine that a robot has to explore the interior of a collapsed building, which has crumbled due to an earthquake, to search for human survivors. It is clearly impossible to have a knowledge of the building's interior geometry prior to the exploration. Thus, the robot must be able to see, with its onboard vision sensors, all points in the building's interior while following its exploration path. In this way, no potential survivors will be missed by the exploring robot. The exploratory path must clearly reflect the topology of the free space, and, therefore, such exploratory paths can be used to guide future robot excursions (such as would arise in our example from a rescue operation).

There are several online computational geometry algorithms for searching or exploring unknown polygons with or without holes, which assume that the visibility region can be determined in a continuous fashion from each point on the path of a robot. Although this assumption is reasonable in the case of a human watchman, it may not be practical in the robotic case for several reasons. First, autonomous robots can carry only a limited amount of onboard computing capability. In the current state of the art, computer vision algorithms that could compute visibility polygons are time consuming. The computing limitations suggest that it may not be practically feasible to continuously compute the visibility polygon along the robot's trajectory. Second, for good visibility, the robot's camera will typically be mounted on a mast. Such devices vibrate during the robot's movement, and, hence, for good precision (which is required to compute an accurate visibility polygon) the camera must be stationary at each view. Therefore, it seems feasible to compute visibility polygons only at a discrete number of points. Hence, in this article, we assume that the visibility polygon is computed at a discrete number of points.

Although the earlier discussion suggests that a robot can only compute visibility polygons at discrete points on its path, it is not clear whether the total cost for a robotic exploration is dominated by the number of visibility polygons that it computes or the length of the path it travels. The computational geometry literature has typically assumed that the cost associated with a robot's physical movement dominates all other associated costs, and, therefore, minimizing the Euclidean length of the path of a robot is considered as the main criterion or the sole criterion for designing motion-planning algorithms. The essential components that contribute to the total cost required for a robotic exploration can be analyzed as follows. Each move will have two associated costs. First, there is the time required to physically execute the move. If we crudely assume that the robot moves at a constant rate, r, during a move, the total time required for motion will be d/r, where d is the total path length

Digital Object Identifier 10.1109/MRA.2008.921542

BY SUBIR KUMAR GHOSH, JOEL WAKEMAN BURDICK, AMITAVA BHATTACHARYA, AND SUDEEP SARKAR

followed by the robot during the exploration. Second, in an exploratory process where the robot has no a priori knowledge of the environment's geometry, each move must be planned immediately prior to the move so as to account for the most recently acquired geometric information. The robot will be stationary during this process, which we assume to take time t_M. Because straight-line paths are the easiest to plan, and since any curvilinear path can be well approximated by straight-line segments, we assume that each move consists of a straight segment. For the reasons outlined previously, we assume that the robot is stationary during each sensing operation, which we assume takes time t_S. Let N_M and N_S be the number of moves and the number of sensor operations, respectively, which are required to complete the exploration of P. Hence, the total cost of an exploration is equated to the total time, T, which is required to explore P: $T(P) = t_M N_M + t_S N_S + d/r$. Now, $(t_M N_M + t_S N_S)$ can be viewed as the time required for computing and maintaining visibility polygons, and it is, indeed, a significant fraction of $T(P)$ because computer vision algorithms consume significant time on modest computers in a relatively cluttered environment. Thus, the goal in this article is to develop an exploration algorithm for a robot that minimizes the number of visibility polygons computed during the exploration of P.

Here, it is assumed that a vision sensor can see any object in P, irrespective of its distance from the sensor, which is not the case in practice. Computer vision range sensors or algorithms, such as stereo or structured-light range finder, can reliably compute scene locations only up to a depth R. The reliability of depth estimates is inversely related to the distance from the camera. Thus, the range measurements from a vision sensor for objects that are far away are not at all reliable. This suggests that it is necessary to restrict visibility polygons by a range distance R. Therefore, only the region of P within R is considered to be visible from the camera of the robot. We refer to the visibility polygon under this range restriction as the restricted visibility polygon. Observe that restricted visibility polygons need not always be closed regions. For exploring P, an online algorithm with a range restriction needs more views than an online algorithm without any range restriction.

This article is organized as follows. In the next section, we discuss the background for our problem on robotic exploration and establish its relation to the art gallery problem. In the section "An Exploration Algorithm," we present an online exploration algorithm without a range restriction and show that the algorithm computes at most $r + 1$ visibility polygons, where r is the total number of reflex vertices in P. The competitive ratio of the algorithm is $(r + 1)/2$. We also show that $r + 1$ visibility polygons are sometimes necessary to see the entire P. In the "An Exploration Algorithm with Range Restriction" section, we present an online exploration algorithm by restricting visibility polygons by a range distance R. The maximum number of visibility polygons that may be needed by this algorithm to explore P of n vertices with h holes is bounded by

$$\left\lfloor \frac{8 \times \text{Area}(P)}{3 \times R^2} \right\rfloor + \left\lfloor \frac{\text{Perimeter}(P)}{R} \right\rfloor + r + h + 1.$$

The competitive ratio of the algorithm is

$$\left\lfloor \frac{8\pi}{3} + \frac{\pi R \times \text{Perimeter}(P)}{\text{Area}(P)} + \frac{(r + h + 1) \times \pi R^2}{\text{Area}(P)} \right\rfloor.$$

Finally, we conclude the article with a few remarks.

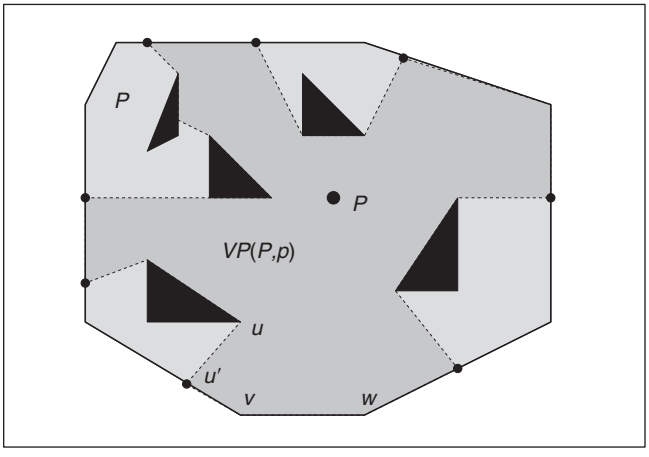

Figure 1. *Visibility polygon of P from a point p.*

Background for Robotic Exploration

In this article, the robot is assumed to be a point. The polygonal environment P is assumed to consist of a boundary polygon, $\mathcal{B}$, populated with h polygonal obstacles, or holes with a total of n vertices. The assemblage of $\mathcal{B}$ and the obstacles is the polygon P (see Figure 1). The free space, $\mathcal{F}$ is the complement of the holes in the interior of P. Prior to the exploration of P, the robot has no knowledge of the geometry of P, the number of edges, or the number of holes. However, we make three assumptions. 1) We assume that the robot can locate its current location relative to a fixed reference configuration. Denote $p = (x, y)$ as the coordinates of the robot relative to this fixed reference. 2) The robot can compute the visibility polygon, $VP(P, p)$ from a viewing point $p \in P$. The visibility polygon $VP(P, p)$ is the set of all points of P visible from p [9] and is bounded by some combination of polygonal edges, partial polygonal edges, and constructed edges as shown in Figure 1. For example, uu' is a constructed edge, $u'v$ is a partial polygonal edge, and vw is a polygonal edge. One of the endpoints of a constructed edge is always a reflex vertex. 3) We assume that the viewing point p and two endpoints of every constructed edge in $VP(P, p)$ are collinear. For example, p, u, and u' are collinear (see Figure 1).

To explore an unknown polygonal environment P, the robot starts from a given position and sees all points of the free space $\mathcal{F}$ incrementally. It may appear that it is enough to see all vertices and edges of P to see the entire free space. However, this is not the case, as shown by the example in Figure 2. In this figure, three views from p_1, p_2, and p_3 are sufficient to see all vertices and edges of P, but the shaded region uvw of $\mathcal{F}$ cannot be seen from these views. This suggests that to see the entire free space the algorithm must ensure that all triangles in the triangulation of P have been explored. Once P is known and triangulated, triangulation of the free space becomes a map of $\mathcal{F}$,

and it can then be used by the robot for its movement between any two locations in $\mathcal{F}$ for future operations.

Suppose $p_1, p_2, \ldots, p_k$ be the viewing points in P such that an optimal exploration algorithm for a point robot has computed visibility polygons from these points, where p_1 is the starting position of the robot. For safety reasons, the next viewing point is always chosen within the region of P that has so far been explored by the robot. Therefore, 1) $\bigcup_{i=1}^{k} VP(P, p_i) = P$, 2) $p_i \in VP(P, p_j)$ for some $j < i$, and 3) k is minimum. So, P can be guarded by placing stationary guards at $p_1, p_2, \ldots, p_k$. So, the exploration problem for a point robot is the art gallery problem for stationary guards [14], with the additional constraint 2). Hence, our exploration algorithms for a point robot are approximation algorithms for this variation of the art gallery problem (when P is not known a priori), which also seems to be nondeterministic polynomial time-hard. For the standard art gallery problem (i.e., P is known a priori and constraint 2) is omitted), Ghosh [8] proposed approximation algorithms for minimum vertex and edge-guard problems for polygons with or without holes that run in $O(n^5 \log n)$ time and yield solutions at most $O(\log n)$ times the optimal.

In the standard art gallery problem, guards can be placed anywhere inside P, and, therefore, there may be guards that cannot be seen by any other guard (see Figure 3). We know that $\lfloor n/3 \rfloor$ stationary guards are sufficient and sometimes necessary for guarding P, which contains no holes [14]. Suppose the guards $g_1, g_2, \ldots, g_k$ are placed in P for security reasons in a such way that each guard g_i for $i > 1$ is visible at least from one other guard g_j for $i < j$, then $\lfloor n/3 \rfloor$ guards are not sufficient as shown in Figure 4. This figure also shows that $\lfloor n/2 \rfloor - 1$ guards are necessary. It has been proved by Hernandez-Penalver [11] that $\lfloor n/2 \rfloor - 1$ guards are not only necessary but also sufficient. If P contains h holes, we know that $\lfloor n+h/3 \rfloor$ stationary guards are sufficient and sometimes necessary for the standard art gallery problem [14] (see Figure 5). If the guards also have to satisfy the visibility constraint between them as stated previously, then $\lfloor n+h/3 \rfloor$ guards are not sufficient as shown in Figure 6. We feel that $\lfloor n+2h/3 \rfloor$ guards are sufficient for this problem.

A watchman route in a polygon is a polygonal path such that every point of the polygon is visible from some point on the path [13], [14]. It can be seen that the path of a robot produced by our exploration algorithm is a watchman route inside P. This path can also be used as an inspection path for autonomous inspection of subsequent traversal [4], [5]. Note that after the viewing points are chosen by our exploration algorithm, the length of the inspection path can be made shorter by connecting these viewing points through shorter paths [5], [15].

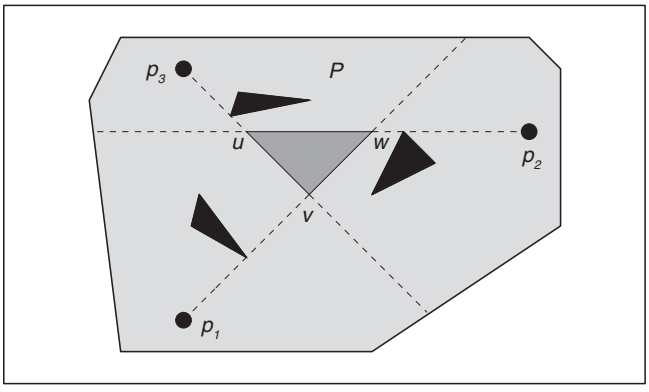

Figure 2. *Three views are enough to see all vertices and edges of P but not the entire region.*

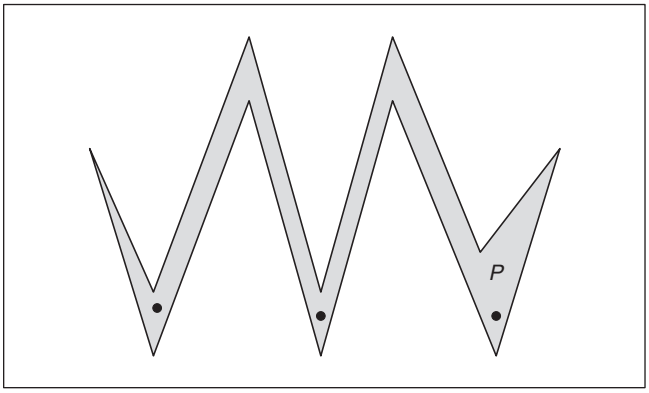

Figure 3. *Three guards can see the entire P, but they are not mutually visible.*

An Exploration Algorithm

In this section, we describe an algorithm that a point robot can use to explore an unknown polygonal environment and guarantee that the entire free space $\mathcal{F}$ has been seen by the robot [10]. We show that the algorithm computes at most $r + 1$ visibility polygons, where r is the total number of

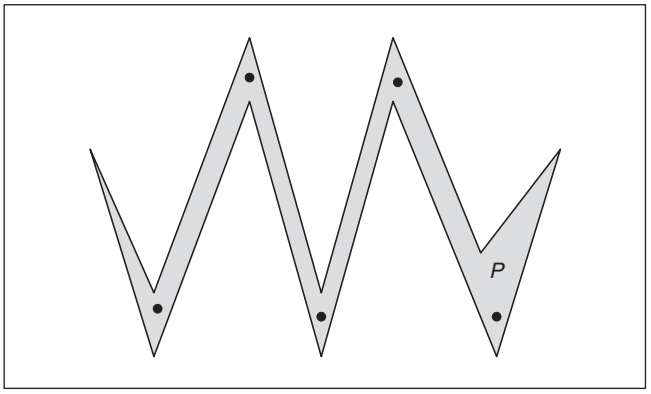

Figure 4. *Two more guards are required to ensure visibility between guards.*

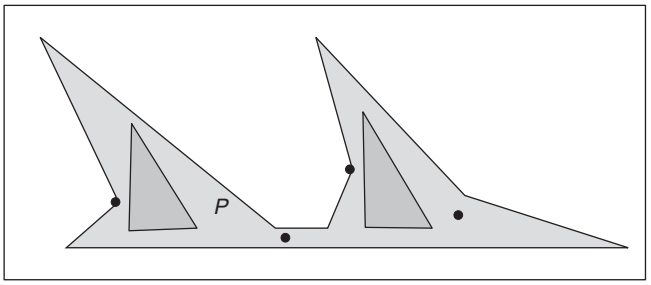

Figure 5. *Four guards can see the entire P, but they are not mutually visible.*

reflex vertices in P containing h holes. We also show that $r + 1$ visibility polygons are sometimes necessary to see the entire $\mathcal{F}$.

The algorithm proceeds as follows. The robot starts at any initial location, p_1, where it determines $VP(P, p_1)$. Using the visible vertices of P in $VP(P, p_1)$, the robot triangulates as much of the $VP(P, p_1)$ as possible. Let this triangulation be denoted T_1. The robot then executes a forward move to the next viewing point p_2. For safety reasons, forward moves are always restricted to the current visibility polygon. Then, it computes the next visibility polygon $VP(P, p_2)$. The region common to

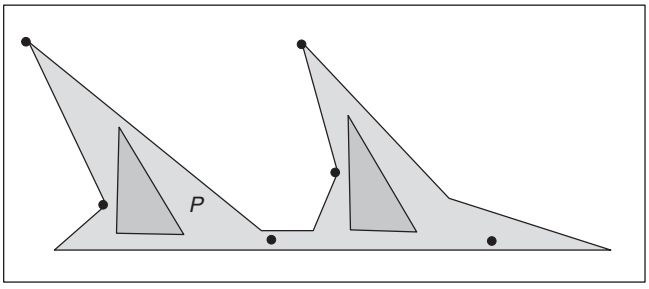

Figure 6. *Two more guards are required to ensure visibility between guards.*

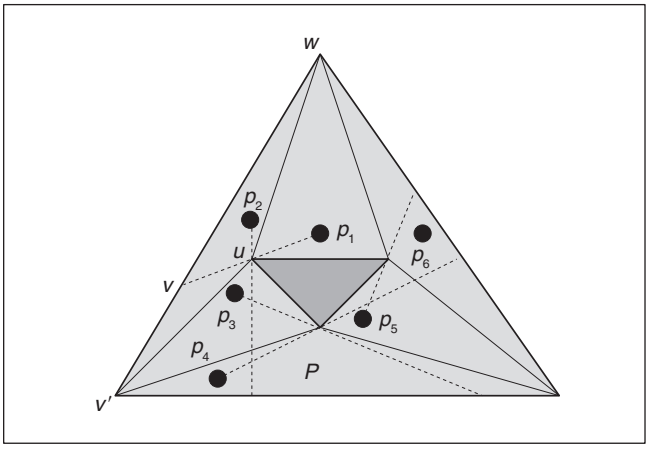

Figure 7. *The algorithm needs $n + 2h - 2$ views; i.e., six views.*

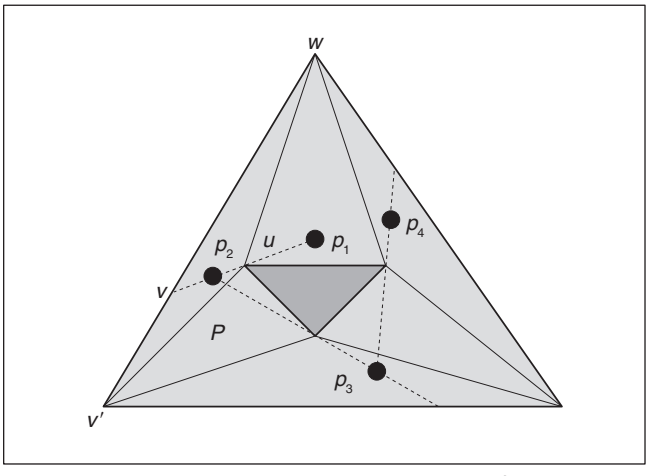

Figure 8. *The algorithm needs $r + 1$ views; i.e., four views.*

The robot must be able to see, with its onboard vision sensors, all points in the building's interior while following its exploration path.

$VP(P, p_2)$ and T_1 is removed. The remaining free space in $VP(P, p_2)$ is triangulated, and the total triangulation T_2 is updated. So, T_2 represents the map of $\mathcal{F}$ explored so far.

To describe the algorithm in more detail, assume that the robot is beginning the ith step of the exploration procedure. Let T_{i-1} denote the cumulative triangulation that has been established prior to step i. Let uv be a constructed edge of $VP(P, p_{i-1})$, where u is the vertex of P and v is a point on a polygonal edge of P. The next viewing point p_i can be chosen from the triangle uvw formed by the constructed edge uv, the boundary edge uw of T_{i-1}, and the partial polygonal edge wv. Figure 7 shows that p_2 is chosen from the triangle uvw. If there is no constructed edge of $VP(P, p_{i-1})$, the robot executes a backward move. The backward move is repeated until a constructed edge uv is located for some $VP(P, p_j)$. A point from the triangle uvw can be chosen as p_i, as stated earlier. Note that if both u and v of a constructed edge happen to be the vertices of P, then w and v become the same and, therefore, p_i is a point on uw.

Once p_i is chosen, the robot computes $VP(P, p_i)$ and then removes the region common to $VP(P, p_i)$ and T_{i-1} from $VP(P, p_i)$. If this operation splits $VP(P, p_i)$ in several disjoint parts, the part containing p_i is chosen as $VP(P, p_i)$. As a result, $VP(P, p_i)$ contains a portion of $\mathcal{F}$, which is yet to be triangulated. It can be viewed as if $VP(P, p_i)$ has been computed by treating T_{i-1} as an opaque region. From now on, whenever we refer to $VP(P, p_i)$, it means that $VP(P, p_i)$ is the connected unexplored region of $\mathcal{F}$ that is visible from p_i and contains p_i.

After computing $VP(P, p_i)$, the interior of $VP(P, p_i)$ is again triangulated by connecting only the vertices of P that lie in $VP(P, p_1)$. The vertices of the triangulation may also include vertices from previously triangulated regions. Let T_i' denote the triangulation of the newly viewed free space in $VP(P, p_i)$. Hence, at the end of step i, the total free space triangulated is $T_i = T_{i-1} \cup T_i'$. In the following lemma, we show that T_i' is not empty.

Lemma 1: *There exists at least one triangle in T_i'.*

Proof: By definition, the next viewing point p_i lies on or outside the boundary of T_{i-1}, and it belongs to a previously computed visibility polygon. Without loss of generality, we assume that p_i is a point of $VP(P, p_{i-1})$. We know that p_i is chosen from the partial visible triangle uvw formed by a constructed edge uv of $VP(P, p_{i-1})$, the corresponding boundary edge uw of T_{i-1}, and the partial polygonal edge wv. Figures 7 and 8 show that p_2 is chosen from the triangle uvw. So, the boundary of $VP(P, p_i)$ consists of the edge uw and a chain of polygonal and constructed edges connecting u and w. Since

this chain makes a turn of 180° or more with respect to p_i, there exists a vertex v' of P on the chain such that $uv'w$ is a triangle inside $VP(P, p_i)$. Hence, $uv'w$ belongs to T'_i. The same argument holds if $p_i \in uw$. ■

Corollary 1: *The viewing point p_i belongs to T'_i and $VP(P, p_i)$ removes at least one constructed edge of $VP(P, p_{i-1})$.*

The above lemma suggests that every time a view is taken, at least one new triangle is explored. From the acquired view $VP(P, p_i)$, the robot establishes a list of constructed edges $(C_{i,1}, C_{i,2}, \ldots, C_{i,c_i})$ on the boundary of the current visibility polygon. These edges help to define the partial visible triangle uvw as stated earlier. Observe that the choice of the viewing point inside such triangles can play a major role in deciding the number of visibility polygons that are required to see the unexplored free space as shown in Figures 7 and 8. Therefore, for subsequent forward exploratory moves, choose a point $z_{i,j}$ $(j = 1, \ldots, c_i)$ on $C_{i,j}$. Choose one of the $z_{i,j}$, for example, $z_{i,1}$, to be the next viewing point, p_{i+1}, and recursively apply this procedure. Hence, the algorithm is a depth-first search of the unexplored free space. When the entire unexplored territory associated with $z_{i,1}$ has been explored, then choose another viewing point at level i, say $z_{i,2}$, and continue. Note that while choosing $z_{i,2}$ as the next viewing point, the corresponding constructed edge $C_{i,2}$ must lie, partially or totally, on or outside the boundary of the free space triangulated so far (called an unexplored constructed edge). Otherwise, it is considered that $C_{i,2}$ has been explored. The algorithm terminates when all the constructed edges of $VP(P, p_1)$ have been explored recursively. In the following lemma, we show that the algorithm has explored $\mathcal{F}$ completely.

Lemma 2: *When all the constructed edges of $VP(P, p_1)$ have been explored recursively, the algorithm has explored the entire free space $\mathcal{F}$.*

Proof: Assume on the contrary that all the constructed edges of $VP(P, p_1)$ have been explored recursively, but there exists a point $z \in \mathcal{F}$, which has not been seen by the algorithm. Consider a path Q inside P connecting p_1 to z. If no such path exists, then z does not lie in the free space $\mathcal{F}$, which contradicts the assumption that $z \in \mathcal{F}$. So, we assume that such a path Q exists. Since p_1 lies in the triangulated region, starting from p_1, Q must intersect at least one boundary edge uv of the triangulation before reaching z. It means that there exists an unexplored constructed edge bounded by u or v, which contradicts the fact that all the constructed edges of $VP(P, p_1)$ have been explored recursively. Therefore, the entire path Q lies inside the triangulation of P. Hence, z has been seen by the algorithm. ■

In the following, we present the major steps of the exploration algorithm for computing the set S of viewing points.

Step 1: $i := 1$; $T(P) := \emptyset$; $S := \emptyset$; Let p_1 denote the starting position of the robot.

Step 2: Compute $VP(P, p_i)$; Construct the triangulation $T'(P)$ of $VP(P, p_i)$; $T(P) := T(P) \cup T'(P)$; $S := S \cup p_i$;

Step 3: While $VP(P, p_i) - T(P) = \emptyset$ and $i \neq 0$ *then* $i := i - 1$;

Step 4: *If* $i = 0$ *then goto* Step 7;

Step 5: *If* $VP(P, p_i) - T(P) \neq \emptyset$ *then* choose a point z on any constructed edge of $VP(P, p_i)$ lying outside $T(P)$;

Step 6: $i := i + 1$; $p_i := z$; *goto* Step 2;

Step 7: Output S and $T(P)$;

Step 8: Stop.

In the following lemma, we prove the upper bound on the number of views that may be required by our exploration algorithm to see the entire free space.

Lemma 3: *The exploration algorithm computes at most $r + 1$ views, where r is the total number of reflex vertices in P.*

Proof: Let u_1v_1, u_2v_2, ..., u_jv_j be the constructed edges generated by the algorithm during exploration. Since one endpoint of every constructed edge is a reflex vertex, we assume that $u_1, u_2, \ldots, u_j$ are reflex vertices. If $u_1, u_2, \ldots, u_j$ are different vertices, then $j \leq r$. So, the exploration algorithm can take at most r views after the initial view at the starting position. Consider the other situation when $u_1, u_2, \ldots, u_j$ are not different vertices. Assume that $u_1, u_2, \ldots, u_{i-1}$ are different vertices, and u_i is the same as u_k where $i < j$ and $1 \leq k \leq i - 2$. We know that the algorithm will choose a viewing point on u_iv_i before any viewing point is chosen on u_kv_k. Let $T(P)$ denote the region of the free space triangulated so far by the algorithm before a viewing point (say, z) is chosen on u_iv_i. Note that $u_1, u_2, \ldots, u_{i-1}$ are vertices of $T(P)$. Since the exploration algorithm uses the depth-first search method, the algorithm explores the entire free space recursively lying outside $T(P)$ that can be reached from z. Let $T'(P)$ be the triangulated region of the newly explored region starting from z. Since u_iv_i and u_kv_k share the same vertex by assumption, there exists a path in the free space from z to every point of u_kv_k and, therefore, entire u_kv_k lies inside $T(P) \cup T'(P)$. So, at most one viewing point can be chosen by the exploration algorithm from those constructed edges that share the same reflex vertex. Hence, the exploration algorithm can take at most r views after the initial view at the starting position.

The correctness of the exploration algorithm and the completeness of the exploration follow from Lemma 1 and Lemma 2, respectively. Lemma 3 provides the upper bound on the computational complexity of the exploration. We summarize the result in the following theorem. ■

Theorem 1: *The exploration algorithm correctly explores the entire polygonal environment by computing at most $r + 1$ views, where r is the total number of reflex vertices in P.*

Let us now compare the performance of our algorithm with that of an optimal exploration algorithm. Consider a spiral polygon without any hole as shown in Figure 9. It can be seen from the figure that no exploration algorithm, starting from p_1, can explore the entire spiral polygon in less than $r + 1$ views. Figure 10 also shows that $r + 1$ views are necessary to explore the entire polygon, which contains one hole. On the

There are several online computational geometry algorithms for searching or exploring unknown polygons with or without holes.

other hand, consider a star-shaped polygon without any hole as shown in Figure 11. In this figure, two views are enough to see the entire star-shaped polygon because the robot takes the first view at the given starting position p_1, and then it takes the second view from the star-point p_2. On the other hand, our algorithm first takes a view from p_1 and, then, takes r views to remove all constructed edges of $VP(P, p_1)$. This example shows that this is the worst performance of our algorithm (i.e., the competitive ratio is $(r + 1)/2$) with respect to the performance of an optimal exploration algorithm. We have the following theorem.

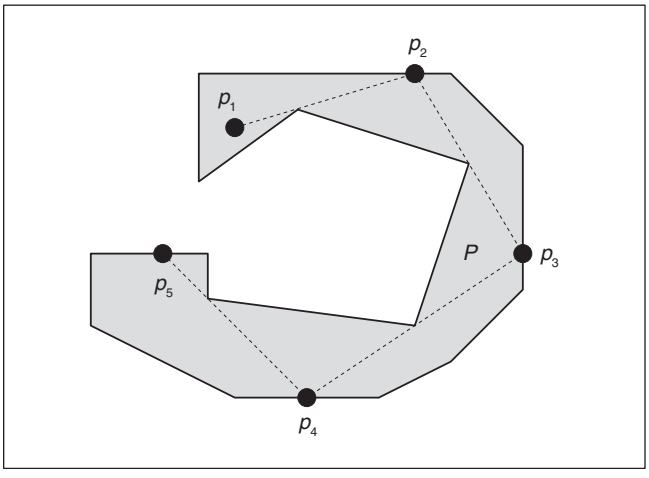

Figure 9. *Any exploration algorithm needs at least $r + 1$ views for exploring the spiral polygon.*

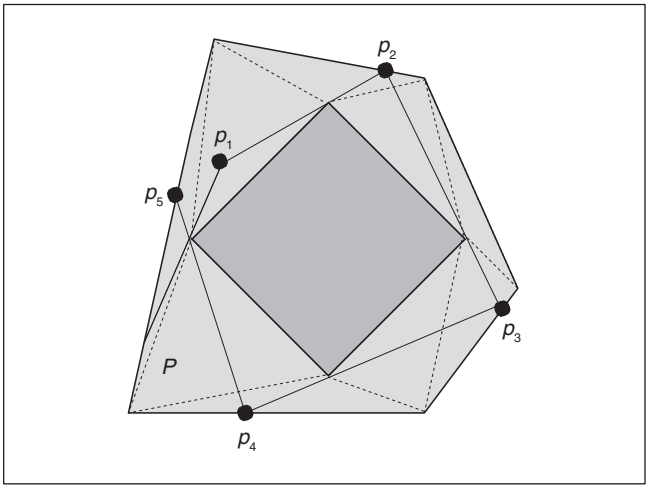

Figure 10. *Any exploration algorithm needs at least $r + 1$ views to explore the polygon with one hole.*

Theorem 2: *The competitive ratio of the exploration algorithm is $(r + 1)/2$, where r is the total number of reflex vertices in P.*

An Exploration Algorithm with Range Restriction

In this section, we present our exploration algorithm by restricting the visibility polygons by a range distance R (see Figure 12) and establish an upper bound for its competitive ratio [2]. This algorithm follows the same approach as in the previous algorithm. We start with the following observation.

Theorem 3: *Let D be the longest line segment that can lie completely inside a polygonal environment P. If $R \geq D$, then the exploration algorithm in the previous section can be used to explore P using restricted visibility polygons.*

Proof: The proof follows from the fact that if $R \geq D$ any visibility polygon computed by the exploration algorithm is the same as the restricted visibility polygon. ■

Let us consider the other case when $R < D$. Let $RVP(z)$ denote the restricted visibility polygon computed from a point z. Observe that a restricted visibility polygon may not

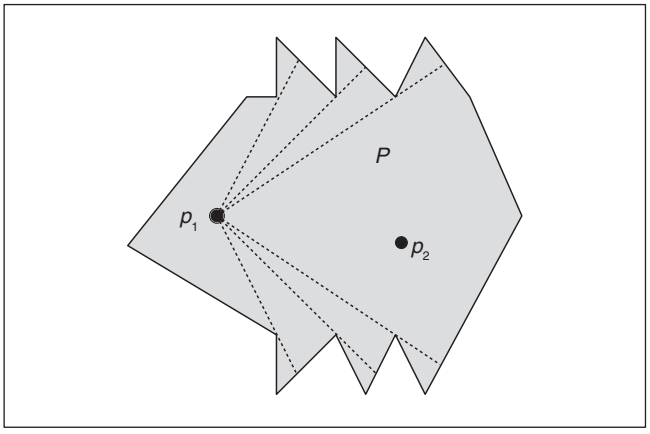

Figure 11. *The star-shaped polygon can be explored in two views.*

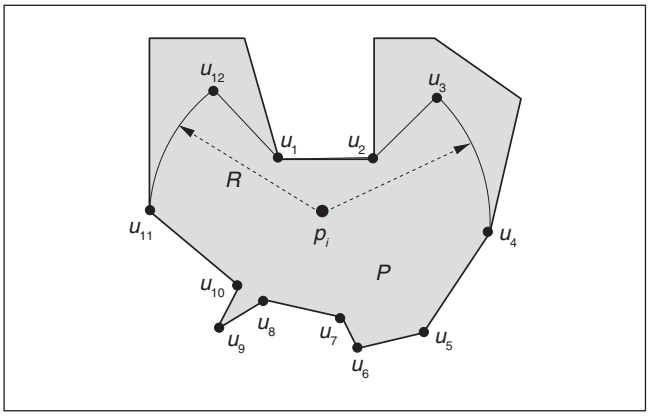

Figure 12. *Vertices of restricted visibility polygon from p_i with range R are $u_1, u_2, \ldots, u_{12}$.*

be closed (see Figure 13), and its boundary can have circular arcs in addition to constructed and polygonal edges. So, it is necessary to take another view from some point in the current restricted visibility polygon to see more of P. The process can be repeated until the union of these restricted visibility polygons covers P.

In the following, we present an algorithm that a point robot can use to explore P by starting from an internal point p_1 and using restricted visibility polygons (see Figure 13). Our algorithm is somewhat similar to the Bug1 algorithm [3], [12] and the coverage algorithm [1], where a point robot moves straight to the boundary of P and then follows the boundary of P. However, our algorithm chooses viewing points arbitrarily close to the boundary of P rather than on the polygonal edges to avoid collision. In the sequel, whenever a viewing point is said to be on a polygonal edge, it implies that the viewing point is arbitrarily close to the polygonal edge.

Let CP_i denote the region of P so far visible from the robot. The robot initializes its polar coordinate system by setting its origin at p_1. So, the coordinates of the boundary points of CP_i are with respect to p_1. Initially, all edges of P are unmarked to indicate that they are not yet explored by the robot. Starting from p_1, the algorithm chooses viewing points on circular arcs until a point u on the boundary of P is visible from the robot. Assume that u belongs to a hole H_i in P. Then, the robot chooses viewing points along the boundary of H_i until all edges of H_i are visible from the robot. Similarly, the robot explores the remaining holes and the outer boundary of P one by one. Finally, the algorithm terminates when the entire free space of P is explored.

Step 1: Compute $RVP(p_1)$; $i := 1$; $CP_i := RVP(p_i)$; *If* the boundary of CP_i consists of only polygonal edges *then goto* Step 8;
Step 2: *While* the boundary of CP_i consists of only circular arcs *do* (see Figure 13)
Step 2a: Choose a point z on any circular arc of CP_i; $i := i + 1$; $p_i := z$;
Step 2b: Compute $RVP(p_i)$; $CP_i := CP_{i-1} \cup RVP(p_i)$;
Step 3: *If* p_i is not a point of a polygonal edge *do* (see Figure 14)
Step 3a: Let z be the furthest point of p_i on the boundary of CP_i that belongs to a polygonal edge; $i := i + 1$; $p_i := z$;
Step 3b: Compute $RVP(p_i)$; $CP_i := CP_{i-1} \cup RVP(p_i)$;
Step 3c: Mark those edges of P that are totally inside CP_i;
Step 4: *While* p_i is a point of an unmarked polygonal edge *do* (see Figure 15)
Step 4a: Starting from p_i, traverse the boundary of $RVP(p_i) - CP_{i-1}$ along polygonal edges until a point z is located, which is a starting point of a circular arc or a constructed edge; $i := i + 1$; $p_i := z$;
Step 4b: Compute $RVP(p_i)$; $CP_i := CP_{i-1} \cup RVP(p_i)$;
Step 4c: Mark those edges of P that are totally inside CP_i;
Step 5: *If* the boundary of CP_i contains a part uu' of a polygonal edge *then* (see Figure 16)
Step 5a: Take a point z from uu'; $i := i + 1$; $p_i := z$; *goto* Step 4;
Step 6: While the boundary of CP_i consists of only circular arcs and marked edges *do* (see Figure 17)
Step 6a: Choose a point z on any circular arc of CP_i; $i := i + 1$; $p_i := z$;
Step 6b: Compute $RVP(p_i)$; $CP_i := CP_{i-1} \cup RVP(p_i)$;
Step 6c: Mark those edges of P that are totally inside CP_i;
Step 7: *If* there is a circular arc or a constructed edge on the boundary of CP_i *then goto* Step 3;
Step 8: Report viewing points $p_1, p_2, \ldots, p_i$ and Stop.

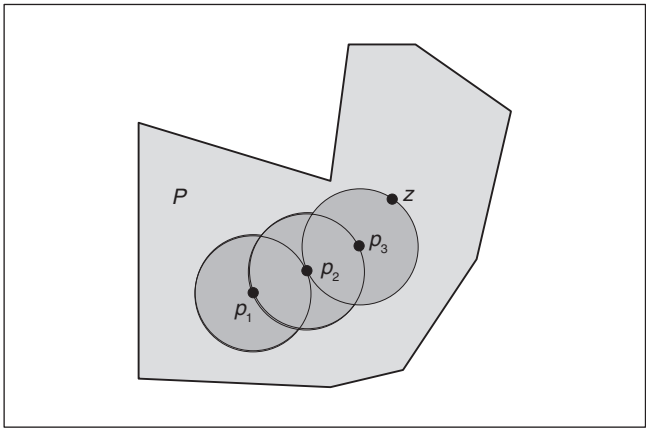

Figure 13. *Restricted visibility polygons are computed in P starting from the initial position p_1.*

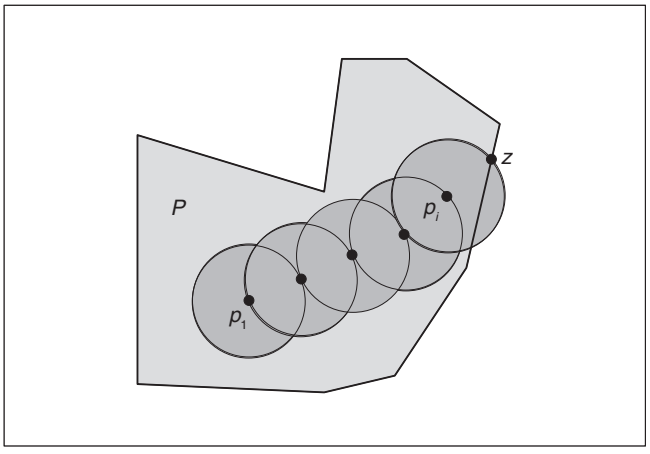

Figure 14. *The current restricted visibility polygon has intersected a polygonal edge for the first time.*

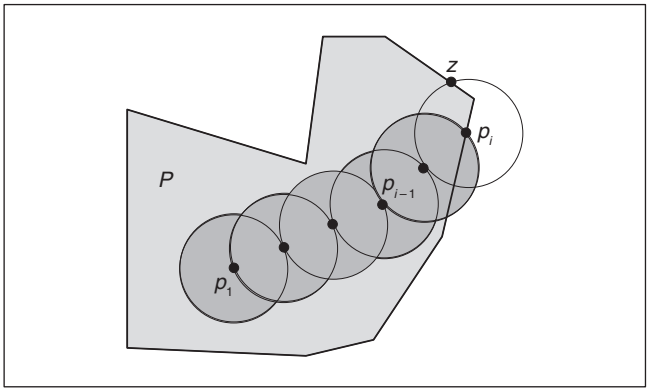

Figure 15. *The robot moves along the boundary of P from p_i until the next viewing point z is located.*

It is open whether an upper bound on the number of views for exploring P under translation can be derived for a convex robot.

Let us prove the correctness of the exploration algorithm. We show that $CP_i = P$ when the algorithm terminates. Assume on the contrary that $P \not\subset CP_i$. Then, there exists a point $p \in P$ and $p \notin CP_i$. If there exists a path from p to p_1 lying inside CP_i, then, p belongs to CP_i, which is a contradiction. Otherwise, any path from p to p_1 must intersect the boundary of CP_i. Since every edge on the boundary of CP_i is a polygonal edge at the time of termination, every path from p to p_1 must intersect an edge of P. So, p does not belong to P, which is a contradiction. Hence, $CP_i = P$ when the algorithm terminates. We have the following theorem.

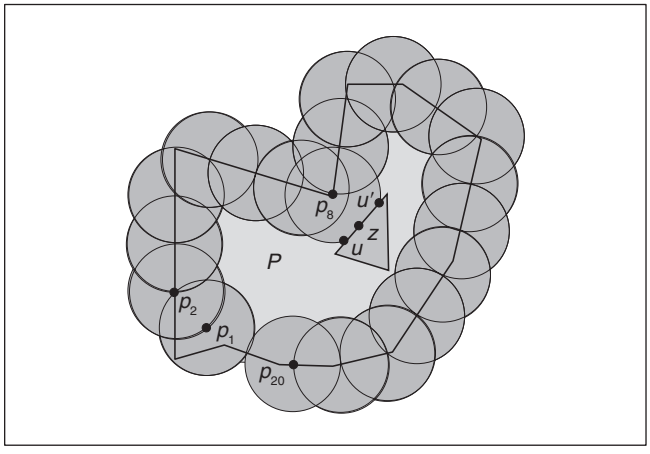

Figure 16. *The robot moves to a point z of uu′ to explore all edges of that hole.*

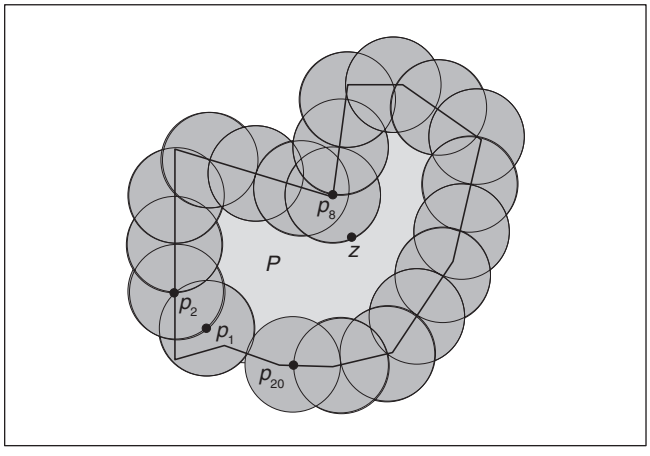

Figure 17. *All boundary edges of P are already explored (i.e., marked).*

Theorem 4: *Using restricted visibility polygons, the exploration algorithm explores the entire polygon P correctly.*

In the following lemma, we establish an upper bound on the number of views required by a robot to explore the region P using this exploration algorithm.

Lemma 4: *If the area, perimeter, number of holes, and number of reflex vertices of P are Area(P), Perimeter(P), h* and *r, respectively, then the number of viewing points required by the exploration algorithm is bounded by*

$$\left\lfloor \frac{8 \times \text{Area}(P)}{3 \times R^2} \right\rfloor + \left\lfloor \frac{\text{Perimeter}(P)}{R} \right\rfloor + r + h + 1.$$

Proof: Place P on a grid where the diagonals of squares is of length $\sqrt{3}R/2$ (see Figure 18). Observe that if the robot takes a view inside a square, it can cover the entire square as every point of the square is within the distance of R from the viewing point. So, the number of squares lying partially and totally inside P gives an upper bound on the number of views required. ■

Consider the squares that lie totally inside P. We know that the area of a square is $(\sqrt{3}R/2\sqrt{2}) \times (\sqrt{3}R/2\sqrt{2}) = 3 \times R^2/8$. Clearly the number of such squares in P is at most $\text{Area}(P)/(3 \times R^2/8) = 8 \times \text{Area}(P)/3 \times R^2$. Since the robot can take at most one view in each square, $8 \times \text{Area}(P)/3 \times R^2$ views can be taken from such squares in P.

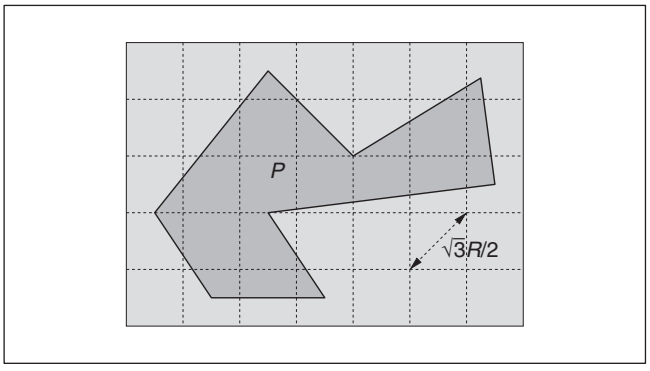

Figure 18. *P is placed on a grid, where the length of diagonals of squares is* $\sqrt{3}R/2$.

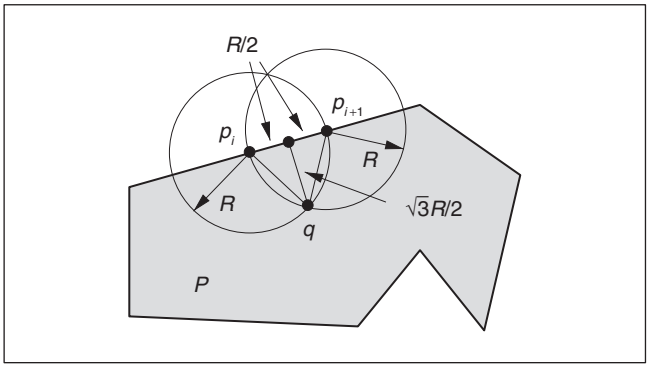

Figure 19. *Viewing points* p_i *and* p_{i+1} *are on the same edge of P.*

Consider the squares that are intersected by edges of P. Instead of counting the number of squares intersected by the edges of P, we show that such intersected squares are covered by views that are taken along the boundary of P. Let p_i and p_{i+1} be two consecutive viewing points on the boundary of P. Assume that both p_i and p_{i+1} lie on the same edge of P (see Figure 19). We know that the distance between them is R if they are intermediate points on the edge. If the distance between them is less than R, then p_{i+1} is a reflex (see Step 4). Consider the case where p_i and p_{i+1} are intermediate points on the same edge. Draw two circles of radius R with p_i and p_{i+1} as centers. It can be seen that any point q, at a distance of at most $\sqrt{3}R/2$ for the segment p_ip_{i+1}, lies within one of the two circles. On the other hand, every point of any square intersected by the segment p_ip_{i+1} lies at a distance of at most $\sqrt{3}R/2$. Therefore, all points of the squares intersected by the segment p_ip_{i+1} lie within $RVP(p_i)$ or $RVP(p_{i+1})$. The same arguments hold if p_{i+1} is a reflex vertex, or p_i and p_{i+1} do not belong to the same edge (see Figure 20). Hence, all points in the squares intersected by edges of P can be seen by at most $\lfloor \text{Perimeter}(P)/R \rfloor + r$ views, where r represents the additional views taken at reflex vertices.

Now, consider the view that sees the boundary of P for the first time (see Step 3). If the view is taken from a square intersected by an edge of P, then we have to add 1 to the bound. Since there can be one such view for every hole and for the outer boundary of P, there can be $h + 1$ additional views in the squares intersected by edges of P. Hence, the maximum number of views that the robot can take to explore P is bounded by $\lfloor (8 \times \text{Area}(P))/(3 \times R^2) \rfloor + \lfloor \text{Perimeter}(P)/R \rfloor + r + h + 1$.

Let us derive the competitive ratio of the exploration algorithm. Since a robot can see in each view at most πR^2 area of P, any exploration algorithm must take at least $\lceil \text{Area}(P)/\pi R^2 \rceil$ views to see the entire P. Hence, the competitive ratio is

$$\frac{\left\lfloor \frac{8 \times \text{Area}(P)}{3 \times R^2} \right\rfloor + \left\lfloor \frac{\text{Perimeter}(P)}{R} \right\rfloor + r + h + 1}{\left\lceil \frac{\text{Area}(P)}{\pi R^2} \right\rceil},$$

which is upper bounded by

$$\left\lfloor \frac{8\pi}{3} + \frac{\pi R \times \text{Perimeter}(P)}{\text{Area}(P)} + \frac{(r + h + 1) \times \pi R^2}{\text{Area}(P)} \right\rfloor.$$

It can be seen that the worst case arises when the number of reflex vertices is large for a given P and R. On the other hand, if R is sufficiently large, the number of views required is $r + h + 1$. We have the following theorem.

Theorem 5: *The competitive ratio of the exploration algorithm for a point robot with restricted visibility polygons is bounded by*

$$\left\lfloor \frac{8\pi}{3} + \frac{\pi R \times \text{Perimeter}(P)}{\text{Area}(P)} + \frac{(r + h + 1) \times \pi R^2}{\text{Area}(P)} \right\rfloor.$$

Conclusions

Our exploration algorithm for a point robot in the "An Exploration Algorithm" section chooses the next viewing point on an unexplored constructed edge. It can be seen from Figure 8 that the viewing point p_4 has been chosen, as the triangle containing p_4 is not totally visible from p_1 or p_3. However, this triangle is totally visible jointly from p_1 and p_3. This observation suggests that if partially visible triangles associated with unexplored constructed edges are taken into consideration while triangulating the current visibility polygon, it may be possible to reduce the number of viewing points by one for every hole that gives the tighter bound $r + 1 - h$. However, it is not clear how to combine these partially visible triangles correctly to generate a triangulation connecting only the vertices of the polygon.

Suppose we wish to design an algorithm that a convex robot $\mathcal{C}$ can use to explore an unknown polygonal environment P (under translation) following a similar strategy of the point robot as stated in the "An Exploration Algorithm" section: let x be the point of $\mathcal{C}$ corresponding to the position of the visual sensor of the robot (see Figure 21). This means that the region

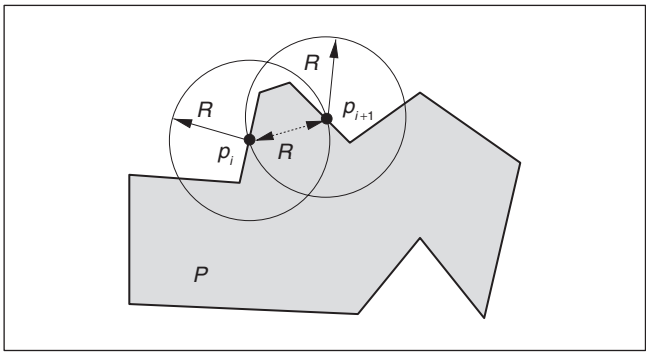

Figure 20. *Viewing points p_i and p_{i+1} are on different edges of P.*

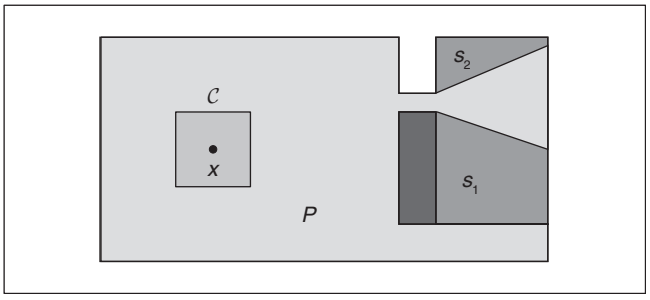

Figure 21. *Two parts s_1 and s_2 of P cannot be explored by $\mathcal{C}$.*

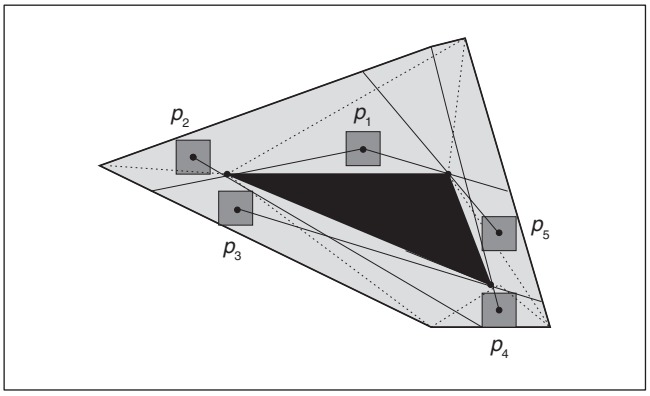

Figure 22. *More than $r + 1$ views are required for exploration.*

visible from the current position of $\mathcal{C}$ is always the visibility polygon computed from x. Observe that since all points of P may not be reachable by $\mathcal{C}$, some parts of P may remain unexplored as shown in Figure 21.

Let p_1 be the starting position of x (see Figure 22). Initially, $\mathcal{C}$ computes $VP(P, p_1)$. Then, it computes the Minkowski sum of C and $VP(P, p_1)$ under translation of C, taking x as the reference point [6], [7]. From the Minkowski sum, constructed edges of $VP(P, p_1)$ are located that can be touched by C, which can be called *eligible* constructed edges. Then, $\mathcal{C}$ moves toward an eligible constructed edge until it touches some point on that edge. Let p_2 be the corresponding position of x. Then, $\mathcal{C}$ computes $VP(P, p_2)$. Using this strategy, $\mathcal{C}$ can explore P but that requires more than $r + 1$ views as shown in Figure 22, because the same reflex vertex is an endpoint of more than one eligible constructed edge. So, the upper bound on the number of views for a convex robot is more than $r + 1$. It is open whether an upper bound on the number of views for exploring P under translation can be derived for a convex robot.

Acknowledgments

The extended abstract of this article was reported in two parts [2]. In [10], we claimed that the point robot computes at most $r + 1 - h$ views to explore the entire free space. However, it has been pointed out that our bound mentioned in [10] is not correct. We have now modified the algorithm by positioning the next viewing point on a constructed edge. This restriction ensures that the algorithm computes at most $r + 1$ views.

The authors would like to thank Sudeb Prasant Pal and Partha Goswami for their helpful comments.

Keywords

Polygon exploration, online algorithms, point robot, competitive ratio, art gallery problem, discrete visibility, visibility polygon.

References

[1] E. U. Acar and H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 345–366, 2002.

[2] A. Bhattacharya, S. K. Ghosh, and S. Sarkar, "Exploring an unknown polygon environment with limited visibility," in *Proc. Int. Workshop Computational Geometry and Applications*, May 2001, pp. 640–648.

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005.

[4] P. Colley, H. Meijer, and D. Rappaport, "Motivating lazy guards," in *Proc. 7th Canadian Conf. Computational Geometry*, 1995, pp. 121–126.

[5] T. Danner and L. E. Kavraki, "Randomized planning for short inspection paths," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, 2000, pp. 971–976.

[6] P. K. Ghosh, "A solution of polygon containment, spatial planning, and other related problems using minkowski operations," *Comput. Vision, Graphics Image Process.*, vol. 49, no. 1, pp. 1–35, 1990.

[7] P. K. Ghosh, "A unified computational framework for minkowski operations," *Comput. Graph.*, vol. 17, no. 4, pp. 357–378, 1993.

[8] S. K. Ghosh, "Approximation algorithms for art gallery problems," in *Proc. Canadian Information Processing Society Congr.*, 1987, pp. 429–434.

[9] S. K. Ghosh, *Visibility Algorithms in the Plane*. Cambridge, UK: Cambridge Univ. Press, 2007.

[10] S. K. Ghosh and J. W. Burdick, "An online algorithm for exploring an unknown polygonal environment by a point robot," in *Proc. 9th Canadian Conf. Computational Geometry*, 1997, pp. 100–105.

[11] G. Hernandez-Penalver, "Controlling guards," in *Proc. 6th Canadian Conf. Computational Geometry*, 1994, pp. 387–392.

[12] V. Lumelsky and A. Stepanov, "Path planning strategies for point automation moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1, pp. 402–430, 1987.

[13] B. J. Nilsson, "Guarding art galleries—Methods for mobile guards," Ph.D. dissertation, Lund Univ., Sweden, 1994.

[14] J. Urrutia, "Art gallery and illumination problems," *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia, Eds. Amsterdam, The Netherlands: North-Holland, 2000, pp. 973–1023.

[15] P. Wang, "View planning with combined view and travel cost," Ph.D. dissertation, Simon Fraser Univ., Canada, 2007.

Subir Kumar Ghosh is a professor of computer science at the Tata Institute of Fundamental Research, Mumbai, India, and he is a fellow of the Indian Academy of Sciences. He has authored several papers on computational geometry, robot path planning, and geometric graph theory, and he has also written a book *Visibility Algorithms in the Plane*, which was published by Cambridge University Press in 2007. He has worked as a visiting scientist in many reputable universities and research institutes around the world.

Joel Wakeman Burdick is a professor in the departments of mechanical engineering and bioengineering at the California Institute of Technology. His current research interests include sensor-based robot motion planning, robotic grasping and fixturing, neural prosthetics, and rehabilitation of spinal cord injuries. He is a past recipient of the NSF Presidential Young Investigator award, the Office of Naval Research Young Investigator award, and the Feynman fellowship. He has been a finalist for the best paper award in the IEEE International Conference on Robotics and Automation in 1993, 1999, 2000, and 2005, and he was appointed an IEEE Robotics and Automation Society Distinguished Lecturer in 2003.

Amitava Bhattacharya is currently a research assistant professor in the Department of Mathematics, Statistics, and Computer Science, at the University of Illinois, Chicago. His main area of research is polyhedral combinatorics. He has published papers in many different areas of discrete mathematics.

Sudeep Sarkar is a professor in computer science and engineering at the University of South Florida, Tampa, where he is an Ashford Distinguished Scholar. His research interests include computer and robot vision, biometrics, and nanocomputing. He is the coauthor of the book *Computing Perceptual Organization in Computer Vision*, which was published by World Scientific in 1994. He is also the coeditor of the book *Perceptual Organization for Artificial Vision Systems*, which was published by Kluwer Publishers in 2000.

Address for Correspondence: Subir Kumar Ghosh, School of Computer Science, Tata Institute for Fundamental Research, Mumbai 400005, India. E-mail: ghosh@tifr.res.in.